5-1-2013

An Exploratory System for Collaborative Decision-Making in Community Planning

Aaron Dale Wells
Boise State University

AN EXPLORATORY SYSTEM FOR COLLABORATIVE DECISION-MAKING IN COMMUNITY PLANNING

by

Aaron Dale Wells

A project

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

May 2013

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the project submitted by

Aaron Dale Wells

Project Title: An Exploratory System for Collaborative Decision-Making in Community Planning

Date of Final Oral Examination: 07 May 2013

The following individuals read and discussed the project submitted by student Aaron Dale Wells, and they evaluated their presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Amit Jain, Ph.D.	Chair, Supervisory Committee
Alark Joshi, Ph.D.	Member, Supervisory Committee
Susan Mason, Ph.D.	Member, Supervisory Committee

The final reading approval of the project was granted by Amit Jain, Ph.D., Chair, Supervisory Committee. The project was approved for the Graduate College by John R. Pelton, Ph.D., Dean of the Graduate College.

dedicated to my children; may their dreams grow wings

ACKNOWLEDGMENTS

The author wishes to express gratitude to CRI Advantage for providing me with the time and means to accomplish this work.

ABSTRACT

Community planning problems differ from those of science, technology, and mathematics as they are not solvable with logical-empiricism. Their solutions are influenced by technology, politics, style, economics, as well as the personalities and experience of those individuals collaborating on the solution. Obtaining cooperation of the stakeholders to implement community planning solutions can be cumbersome or simply cause failure in the implementation of plans. Yet, if the stakeholders had a real handle on the cost and benefits the literature suggests that cooperation can evolve.

In this project, we explore building a reliable cost and benefit model for a set of input parameters that may allow a collaborative solution to emerge more easily. Furthermore we hypothesize that in the decision process there is a tipping point between costs related to a decision and its benefits.

In order to test the hypotheses, we have designed and tested a software framework with focus groups that included locally elected officials, economic development specialist, planners, and citizens. The software framework allowed the stakeholders to explore an interactive cost-benefit model, and researchers to collect those interactions and visualize them in real-time. The software framework developed for the study, its set up, and findings based on a focus group study are discussed.

The software framework developed for this study and the included analysis tool provided were shown to be effective in identifying the "tipping-point" moments in the group dynamics.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASPX – Active Server Page

BSU – Boise State University

CSV – Comma Separated Values

GIS – Geographic Information System

HTTP – HyperText Transfer Protocol

HTML – HyperText Markup Language

IRB – Institutional Review Board

HTML5 – The current standard version of HTML

JSON – JavaScript Object Notation

MEF – Microsoft Extensibility Framework

MVVM – Model View View Model

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

TDC – Thesis and Dissertation Coordinator

URL – Universal Resource Locator

XAML – eXtensible Application Markup Language

XAP – XAML Application Package

XML – eXtensible Markup Language

XPath – Not an acronym. A query language for XML.

XSD – XML Schema Definition

CHAPTER 1

INTRODUCTION

1.1 Problem Overview

Cooperation and collaboration between multiple stakeholders is a key indicator of sustainable neighborhoods, cities and regions. Much of this work takes place with individuals and institutions. In this area, the topic of cooperation is of special interest because it forms the foundations of economic, social, and cultural decisions within cities or larger communities.

Several theories exist regarding how individuals' decision making process are influenced by their individual circumstances and communities [8]. Beyond traditional influences, some studies have shown that agent-based modeling can perform a role in local and regional decision making [10]. Where study participants have access to GIS information, it has been shown to be an informative element of the model [5] and especially useful in public works planning collaborations [9].

Research of cooperative behavior within a population of autonomous individuals has led to studies of multi-agent systems and the evolution of cooperation [7]. These studies have focused on modeling human behavior. The proposed study requires participants to have the ability to make decisions and influence outcomes.

Social scientists within Boise State University have hypothesized that beyond the self-interest and reciprocity models, human participants may also demonstrate a

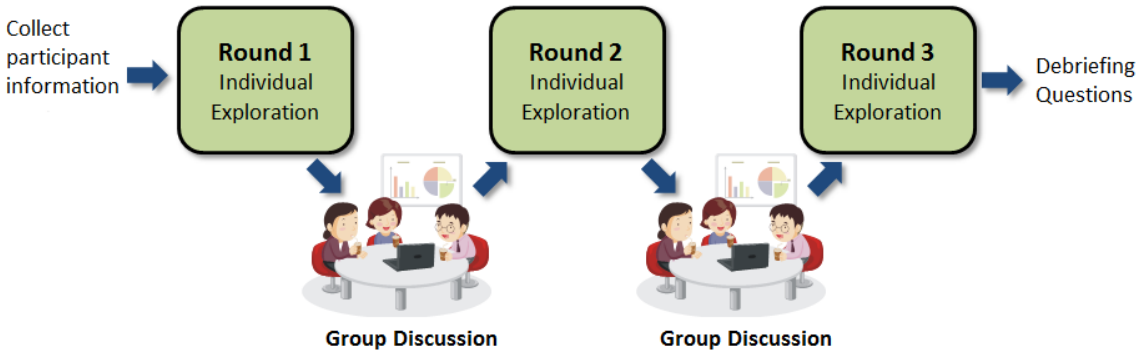

Figure 1.1: A depiction of the structure of the focus group's evaluation

"tipping point" where individuals agree to cooperate even if benefits are not portioned to their own. In order to evaluate this hypothesis, a means of collecting real-time data from live study participants is required. This approach is consistent with recent developments in urban planning efforts focused on increasing public participation and eliciting community input [1].

For this purpose, a hypothetical multi-purpose stadium to be built within the Boise city limits was used as the basis for our pilot. Study participants were to interact with a model of the stadium's impact on the community in terms of traffic, noise, and financial impact. Inputs such as: number of seats, amount of parking, and the number and type of events impact the cost, noise, traffic, and projected economic impact in the model. Participants were able to see how their choices compared to others in their group. Timed and event-based interactions with the model caused data points from the model to be logged in a database for later analysis. The framework was flexible enough to allow rapid reconfiguration of the model and demographic data collected.

In our focus group, each user was allowed 5 minutes to individually explore the

model space. At the end of each round, each user was required to select a set of parameters they felt represented the best available configuration for the stadium. After each round, they participated in a group discussion to ask questions and understand the others' choices. A composite model representing the average of the user's choices was presented to the users. The discussion was followed by another round of exploration, and so on (see Figure 1.1). There were a total of three rounds of discussion. At the end of the discussions, each participant was asked to select their preferred model from the models presented at the end of each round. After the exploration, we conducted debriefing surveys with each participant in separate rooms to obtain qualitative feedback on the process as well as the software framework.

1.2 Experimental Setup

To gain a better understanding about decision making when it comes to large infrastructure projects, we implemented a software framework to collect the quantitative data and provide real-time visualization.

Data gathering was organized into a series of techniques with varying levels of participation in the study:

1. An initial survey with questions pertaining to demographics, participation in local economic development, and some general background questions.

2. After the facilitator presented instructions, individual participants selected from constrained parameterized choices about the proposed multipurpose stadium. These selections were collected in a central data store as they were made and changed. A complete history of each participant's choices was stored.

3. After each round of individual interactions with the software framework, the participants were able to view the choices of other participants. Proctors were able to view the choices as they were being made and changed in real time.

4. After the completion of the exercise, the participants were asked to complete an exit interview with questions that pertain to the factors that influenced the decisions that they made.

For the hypothetical decision making step on this facility, we provided hypothetical data that is based in as much reality as we could for:

1. Facility Inputs: stadium size, number of seats, percentage of covered versus uncovered seating, parking spaces, and the option for additional events.

2. Facility Outputs: tax model (increase in property tax per capita), traffic congestion [reduction of MPH], and effects of noise from the facility.

3. Facility Benefits: hotel and restaurant receipts and overall benefit to the city in terms of 'community pride' with having such venues and attractions.

All personal data was held confidential and the focus group design, survey, and debriefing questions were approved by the Institutional Review Board (IRB).

Each participant was provided choices on the stadium size, seats, parking and events. The noise (in decibels) is presented graphically with a red cloud projecting its reach and intensity. Each participant had the opportunity to provide their property value to see the impact of the stadium choices they would make on their own wallet. This information was not recorded and only provided to help participants with their exploration and decision making on the stadium. This aspect of individual exploration

of scenarios and their impact on one's decisions provides a unique insight into the potential consequences of their actions on the community as a whole.

Focus Group We invited people that we expected would be interested in the development of the downtown stadium. Participants came from the community and represented one or more of the following: policy maker, private sector developer, and potentially concerned citizen. Participants were contacted by phone and then a follow-up email was sent with details of the study purpose, time and location. There were two sessions with four different participants in each session. Each session had representatives from development, economic development, the city of Boise and one "concerned citizen". The first focus group had two women and two men. And the second focus group was comprised of three women and one man. Participants were invited to attend a focus group for one and half hours.

Software Framework Initial demographic information was collected by the software framework and consisted on quantitative as well as qualitative data.

Focus group participants interacted with a web-based model of the stadium's impact on the community. Inputs such as: number of seats, amount of parking, and the number and type of events will impact the cost, noise, traffic, and projected economic impact if such a stadium was to be built. Inputs were constrained to discrete values in order to limit the potential permutations and encourage convergence on a shared solution. Participants could see how their choices compared to others. Timed and event-based interactions with the model were logged in a database.

Later chapters of this document elaborate on the software framework.

CHAPTER 2

FUNCTIONAL REQUIREMENTS

Tipping Point The objective is to find the *tipping point* in a group decision making process. In a group decision the *tipping point* occurs at a point where the group behavior or consensus, seemingly headed toward a particular decision, changes, ever so slightly, and begins changing course. Mathematically it is not at the top or the bottom of the function but at the point of inflection. It is that point where the concave function becomes convex or the convex function becomes concave.

Some claim that a tipping point has occurred on a grand scale in the discussion of global warming [4]. It is asserted that a change of attitude occurred in the general public and a growing number of people have moved from a position of skepticism about the concept of warming to a position of accepting its reality and considering those actions that could/should be taken to alleviate the anticipated problems.

The tipping point is not characterized by unanimity and perhaps not even majority. It is the point where the inevitability changes 'direction'. Nor does the tipping point necessarily signal the inevitability of an outcome. Given the ebb and flow of public discourse, one tipping point can be followed by any number of tipping points. Only after the final decision has been made can we identify the final tipping point.

We believe the tipping point(s) can be discovered through a process of *reverse engineering*. Given the final decision, and if the decision making process has been

documented, it should be possible to retrace the steps backward from the decision to the last turning point and from there to any earlier turning points. We should be able to identify those points in the discussion where the available information and group dynamics changed the direction from "concave" to "convex", i.e. it should be possible to identify all the "points of inflection".

In our pilot, we able to identify the tipping point leading to final convergence for several input parameters. However the number of participants was small, so further tests on other settings would be beneficial. However the tool that we have developed has demonstrated the capability to identify tipping points.

2.1 General

The application shall collect demographic and model information (the data) from users during a facilitated discussion. The current state of the demographic and model information shall be collected in real time at periodic intervals into a database. The current information will be redistributed to allow users to visualize how their model choices compare with those of their peers.

The data shall be collected so that post-discussion analysis may be performed to possibly determine if evidence of a "tipping point" exists. Data shall be collected with absolute and relative time stamp information.

As needed by researchers, the model may be amended. Each facilitated discussion should be viewed as a separate data set.

The system shall allow for simultaneous administration of different models without changing the application.

Multi-media data types (such as geospatial, three-dimensional, images, audio, and

video, etc.) shall be available in the system as either native types or plug-ins to the system. Each data type shall store its own relevant data for later analysis.

2.2 Distribution

The application shall be distributed as a web-browser application over standard HTTP(s) protocol. It shall run in supported web browsers using industry standard browser plug-ins as necessary to provide a compelling user experience.

2.3 Scalability

In order to evaluate the impact of virtual group participation on the outcome of a survey, participants and proctors must be able to observe the choices their peers are making in near real time. This requires that some form of the data from every client be distributed to every other client. As the number of clients grow, the amount of data that must be exchanged grows on the order of the square of the number of clients.

The communication protocol for distribution and access of this system shall be HTTP over the public internet. While this protocol allows near universal delivery and access, it disallows the ability to use a broadcast protocol typically used on private networks to minimize the exchange of client data. Likewise, it also disallows peer-to-peer distribution protocols for the same reasons.

2.4 Flexibility

The system shall be flexible so as to accommodate changing model and demographic requirements, and provide an environment where different types, or evolving definitions for surveys may take place. However, as a boundary within that framework, the following specifications are mandatory:

- Ability to allow study participants to access the application without installing any custom software.
- Ability to collect participant data of the following types
- Multiple choice
- Free-form text
- Interactive Geospatial
- Three-dimensional model
- Complex type (model)
- Ability for developers to extend the data types collected to include other desired types or models without recompiling the core application, and distribute those models dynamically.
- Ability to group and order participant data into meaningful partitions.
- Ability to provide limited workflow between sections.
- Ability to change the collected data without recompiling the application.

- Ability to store the collected data to a data store on a centralized or distributed data repository.
- Ability to have multiple simultaneous configurations hosted on a single server.
- Ability to provide limited interactive analysis of a given configuration to participants.

2.5 Data Collection and Future Analysis

The data store selected for the system should be flexible enough to handle many different survey types without modification. In order to maximize performance and scalability, data should be stored simply, without the overhead of building, maintaining, and enforcing foreign key relationships. Each survey result should be treated as an atomic unit that could be distributed at the time of collection, or during analysis. While detailed analysis of the data will not be a focus of this project, the method of storing the data as well as the storage format selected should lend itself to massively parallel analysis.

CHAPTER 3

TECHNICAL SPECIFICATION

3.1 General

It is desirable that clients be able to access the system without the installation of custom software. However, some of the advanced capabilities of this survey engine will require that a capable plug-in be used to extend the native capabilities of web browsers beyond that currently available in HTML5 capable browsers using off-the-shelf components. Therefore, web clients shall be required to support a capable browser plug-in such as Flash, Java, or Silverlight.

The system shall have three physical application layers: Data Store, Service, and User Interface as shown in Figures 3.1 and 3.2. The User interface shall be divided into four logical tiers: Service Access, Model, View Model, and View. Each layer is dependent upon the layer beneath it.

3.1.1 Out of Scope

User authentication *shall not be addressed* in the survey client application of this system. Any anonymous user shall have access to surveys in this system. User authentication may be effectively wrapped around the survey system by native mechanisms built into the HTTP protocol.

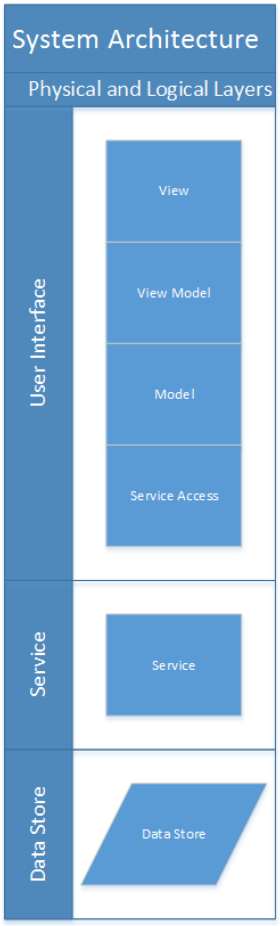

Figure 3.1: Basic Architecture

3.2 Data Store Layer

The data store is required to hold the serialized results of surveys. XML (extensible markup language) shall be used as the serialization format. The exact schema of the serialized XML is dependent upon the configuration of the survey being performed, so shall not have a required XSD.

The data store layer shall also contain transformations required to provide the data to the interactive analysis portion of the user interface.

3.3 Service Layer

The service layer implementation is a web server that hosts the following:

- HTML/ASPX target navigation page(s) for the active survey(s)
- Compiled XAP (XAML execution package) application package
- Compiled XAP extension packages
- Survey configuration file(s)
- Web service for retrieving/sending data with the data store written

3.4 User Interface Layer

The user interface is a Microsoft Silverlight application using the .NET 4.0 framework. Upon loading, the application loads its configuration file as well as any extension packages required. When these are loaded, it begins working through the user workflow and sends survey information to the service layer on a periodic basis.

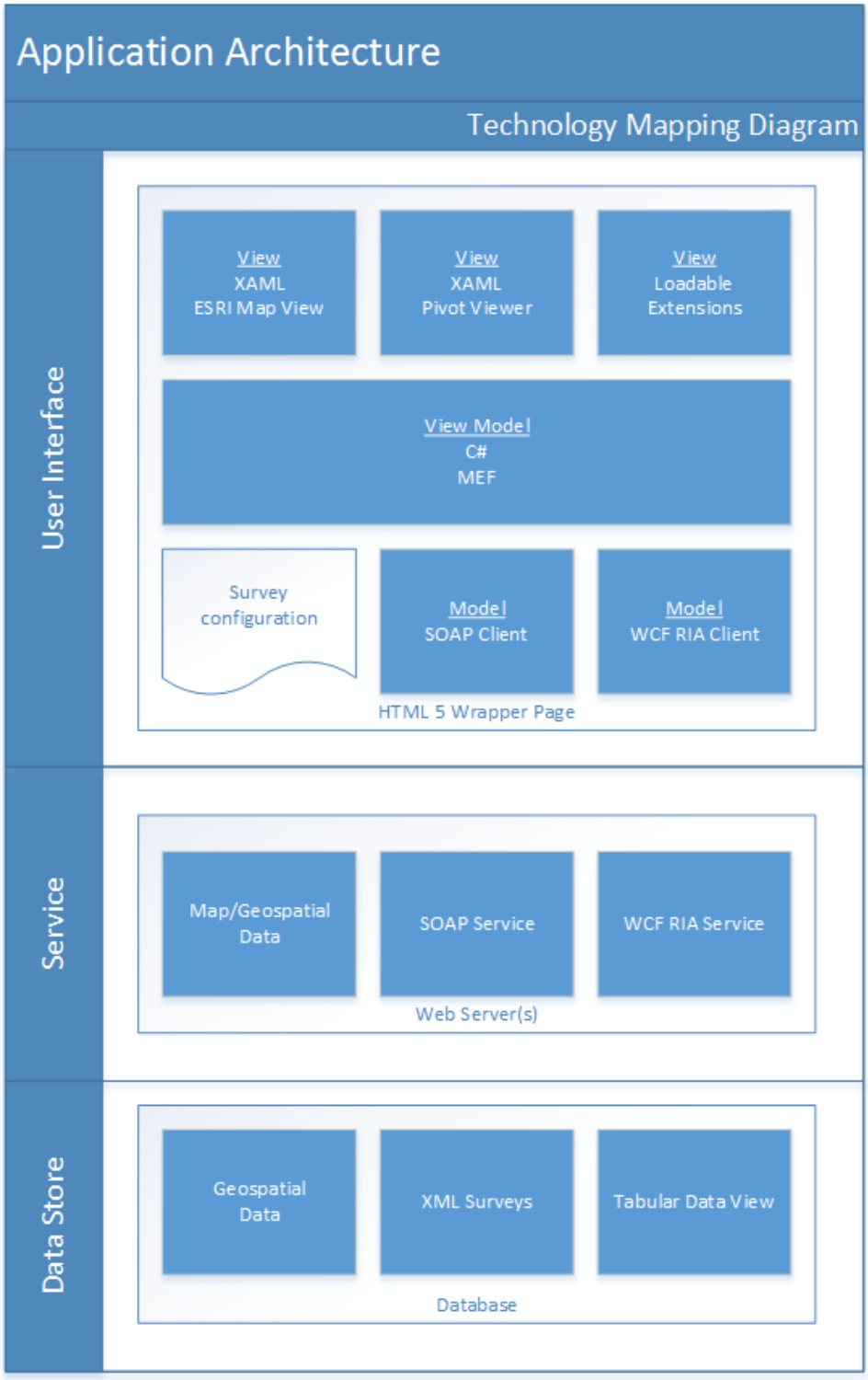

Figure 3.2: Technology and Architecture

The analysis portion of the user interface layer retrieves current survey information from the service layer on a periodic basis.

Service Access Tier This tier is tied to the signature of the services accessed by the application. Together they form an application programming interface.

Model Tier The model tier contains the entities shared with the service layer as well as application logic pertaining to those entities. This application is considered a "rich client" because it makes extensive use of the client to perform processing; and as such has most of the application logic in the model of the client application.

View Model Tier The view model tier contains objects that orchestrate models and coordinate user events with actions on the model. This layer is constructed in such a way as to decouple it from the widgets that the user sees. This application uses the view model layer to dynamically load the configuration file and extensions.

View Tier The view tier is constructed in XAML and has very limited code. Where there is code in this layer, it has a purpose of providing the appropriate user experience.

CHAPTER 4

SYSTEM DESIGN

All design is an exercise in compromise. The design and development of this system could not hope to be any different. While the author believes that the resulting system meets the requirements and specifications outlined, there will certainly be those who would have selected a different solution given the same constraints.

Many of the choices were made with a bias towards expediency as well as functionality; familiar tools and technology were selected over unfamiliar tools with equivalent and equally acceptable capabilities.

4.1 Data Store

The data store for this project may be viewed as a container for of XML documents. Each document is a snapshot of the state of the user selections at a given moment in time. XML itself is not especially efficient as a transport format, but does lend itself to a variety of analysis tools after the data has been collected. Each XML Document contains (at a minimum) a survey identifier, and an instance identifier. Because these documents are completely self-contained, they need not be collocated by survey or by survey instance at the time of data collection. The lack of these constraints would allow the data collection system to scale by using inexpensive hardware and pooled or distributed hardware scaling techniques.

"Big Data" techniques may easily be applied to the data once collected. For example, a Hadoop cluster could be used to analyze the distributed data.

The solution used for this project does not currently employ a distributed architecture for data collection. It uses Microsoft SQL Server as a data repository. The purpose in repurposing a database engine to emulate a file store is that SQL Server contains an excellent built-in XML parsing engine, allowing queries to be built against XML contained in the tables using extensions to standard SQL. While analysis was not the primary focus for the system at this time, the choice of a database engine over a distributed cluster provided similar results with less administration, maintenance, and more flexibility for the limited datasets obtained during testing.

The database used to store the survey instance data consists of only one table, and that table contains only two columns: an auto-incrementing key, and an XML field. In order to improve the performance of the analysis characteristics of the data store, XML indices could be employed on the table. This optimization should be applied to complete data sets to avoid the speed penalty incurred when indexing occurs in a database on insertion.

4.2 Service Interface

One purpose of a service interface is to hide the complexity of the underlying system from the client applications and insulate them from changes. The service contract of this application consists of three parts: client configuration, data collection, and data distribution.

Client Configuration Each survey is accessed via a distinct URL. The webpage found at that location instructs the browser to load the Microsoft Silverlight client

```
<body>
  <div id="silverlightControlHost" style="height: 100%">
    <object data="data:application/x-silverlight-2,"
        type="application/x-silverlight-2" width="100%" height="100%">
      <param name="source" value="ClientBin/SurveyEngine.xap" />
      <param name="onError" value="onSilverlightError" />
      <param name="background" value="white" />
      <param name="minRuntimeVersion" value="5.0.61118.0" />
      <param name="autoUpgrade" value="true" />
      <param name="enableGPUAcceleration" value="true" />
      <param name="initParams" value="survey=DemoConfiguration.XML,
        package0=SurveyEngine.Package.xap"/>
    </object>
    <iframe id="_sl_historyFrame"
      style="visibility: hidden; height: 0px; width: 0px; border: 0px"></iframe>
  </div>
</body>
```

Figure 4.1: Sample HTML code to embed the survey engine into a web page.

and has the configuration details needed by Silverlight to load the XAP file containing the Survey Engine. It also contains additional configuration information used by the Survey Engine to load the survey configuration XML file and any additional packages needed for content types not built into the Survey Engine. Figure 4.1 the body of an HTML page demonstrates a possible configuration:

The param containing the name "initParams" contains the information needed to select the survey configuration as well as dynamically loadable modules required by the survey configuration. The contents of the configuration XML file are determined by the survey question types required. It consists of a serialized survey object. An example of the contents of a configuration file may be found under Appendix A. The general layout hierarchy of the file is as follows:

Survey the root element for the document; a survey has an identifier and a title.

Rounds timed repetitions of various pages of the model; each round contains an identifier, name, description, time allowed to complete the round, and whether or not the data from that round is recorded in the data store.

Pages elements that may or may not be visible during a round; pages contain a title and indicate in what rounds they shall be visible.

Sections a visual grouping for elements.

Wrapped Object – container for survey elements. Each type of survey element has its own XML structure.

Several types of survey elements were created for the pilot. The system is designed to progressively accommodate more elements as needed. Each of the core types are generic types, and therefore may be instantiated using other types. For example, a labeled value of integer and labeled value of string are the same generic type, but are instantiated and validated separately. A few examples are provided here; the source code should be referenced as a more complete documentation.

Client Authentication and Authorization Normal authentication and authorization techniques may be utilized to secure or restrict access to the survey in any of the same ways employed elsewhere for HTTP authentication and authorization. For the purposes of this pilot, no authentication or authorization mechanisms were employed.

Data Collection The client application sends user survey data to the Service Interface via a SOAP web service. The service interface has only one method:

1. AddSurvey(string XMLSurvey) – store the survey information to the repository.

Table 4.1: Example configuration settings

LabeledValueOfInt32	<Title>What is your age?</Title>
LabeledValueOfString	<Title>What is your name?</Title>
MultipleChoiceQuestionOfStringInt32	<Title>What is your gender?</Title> <Choices> <ChoiceOfStringInt32> <Title>Female</Title> <Value>1</Value> </ChoiceOfStringInt32> <ChoiceOfStringInt32> <Title>Male</Title> <Value>2</Value> </ChoiceOfStringInt32> </Choices>

The purpose in using a string rather than XML as the data type for the parameter is that string is an atomic, universally recognized type, while XML is not; this will also allow for future clients to be written that use a different encoding scheme (such as plain text, CSV, or JSON).

4.3 User Application

The user application has two user screens, and a utility screen. When the application initializes, it navigates by default to the home screen. This is the screen used to collect demographic and model data from each user.

The user application application is written using Microsoft Silverlight [6], and requires version 5 of the Silverlight runtime to be installed. The Silverlight runtime is compatible with major browsers for Windows and Macintosh operating systems. The Silverlight client communicates over HTTP with the server using web services.

A treatment of Silverlight and web services is beyond the scope of this document. The author has attempted to follow industry conventions in building this application.

Dependencies The application was built using Microsoft Visual Studio 2010 (service pack 1) and later, Visual Studio 2012. It uses C# as the primary programming language.

The client application requires that a few development SDKs be installed on the development machine:

- ESRI ArcGIS Client – used for map visualizations [3]
- Silverlight Developer Toolkit – prerequisite for the Pivot Viewer [6]
- Silverlight Pivot Viewer – analysis functionality [6]
- GalaSoft MVVM Light – helper types used for the Model-View-View Model patter employed by the application. [2]
- Microsoft Extensibility Framework (MEF) – Formerly a Microsoft provided library for dynamic component instantiation. Included in Microsoft .NET 4.5.

The Silverlight toolkit and MVVM Light components are included in the code repository as NuGet packages, and may be retrieved directly using NuGet. The other packages must be installed. The ESRI ArcGIS Client requires a developer account to access the SDK.

Home Screen The home screen (Figure 4.2) contains the visual representation of the configuration file for the survey. This screenshot shows the survey title (Boise Multi-use Stadium), the name and description for the first round (in the modal dialog

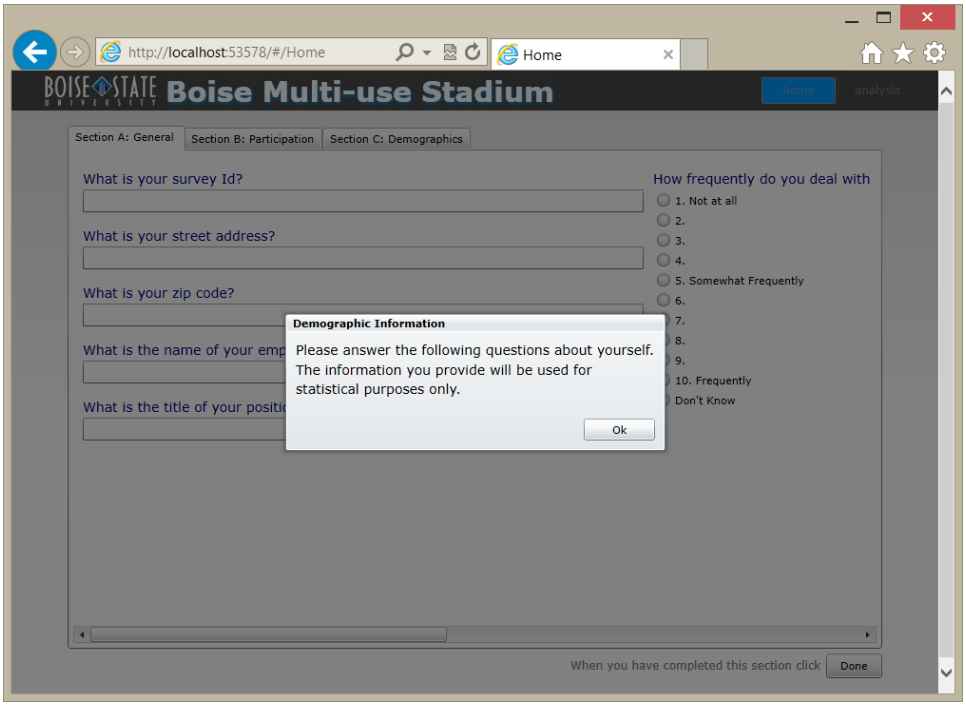

Figure 4.2: Home Screen

box), several pages (shown as tabs), and sections containing a few labeled values and multiple choice questions.

Modeling Screen The modeling screen (Figure 4.3) is the primary workspace for participants in the study. It is also the most complex of the data viewers in the application. This screen contains controls allowing the user to adjust inputs for various model inputs. The model reacts to changes in these inputs and provides both numeric and visual feedback. The relative noise level is the primary visual feedback, while various costs and benefits are represented numerically.

Changes in the model are persisted on a periodic basis to the underlying data store for limited real-time visualization in the analysis screen and more in-depth analysis at a later time.

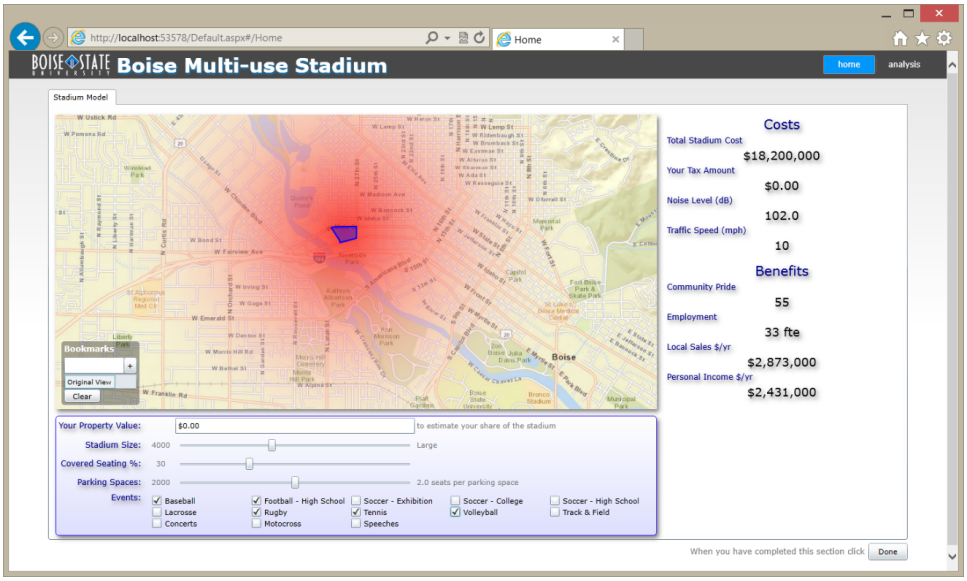

Figure 4.3: Modeling Screen

In the following screen-capture, the location of the proposed stadium is overlaid on the map in blue. The map is fully interactive, allowing pan and zoom operations as well as bookmarks. The red area around the stadium represents the relative effect of peak noise. The user may explore the consequences of various event types, the number and type of seating, and the number of parking spaces. Property value may also be entered, but is not collected; it is used to compute a projected increase in property tax on a theoretical property. Costs and benefits are placed together in the right panel.

Analysis Screen The analysis screen (Figure 4.4) uses the pivot viewer Silverlight component provided by Microsoft (Microsoft n.d.). The viewer is capable of filtering multi-dimensional data by any number of the values in each dimension and displaying a visual representation of the data in a grouped bar graph, or grid layout. Each data element may be identified by one or more labels. The data may be graphically

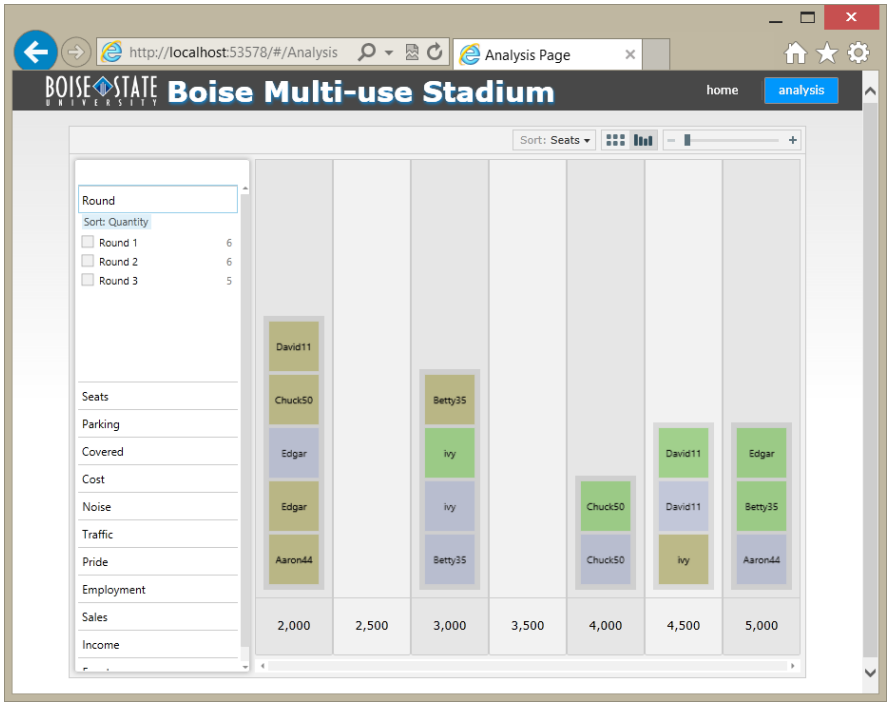

Figure 4.4: Analysis Screen

distinguished by image and/or color.

The interactive nature of the pivot viewer allows a user to add or remove filters and see the data react.

In the survey application, we extend this interactive capability by updating the dataset with the most current values available from the data store. As many users participate in a survey, an observer or other user(s) may see how changes effect individual participants as well as the group.

In our example, the dimensions provided to the pivot viewer were a combination of model inputs and computed values. Each of the three data collection rounds is assigned a different color. Each cell is labeled with the survey identity of the user.

It should be noted that the analysis screen is neither modular, nor adaptive to radically different survey configurations. It depends upon a purpose built pair of

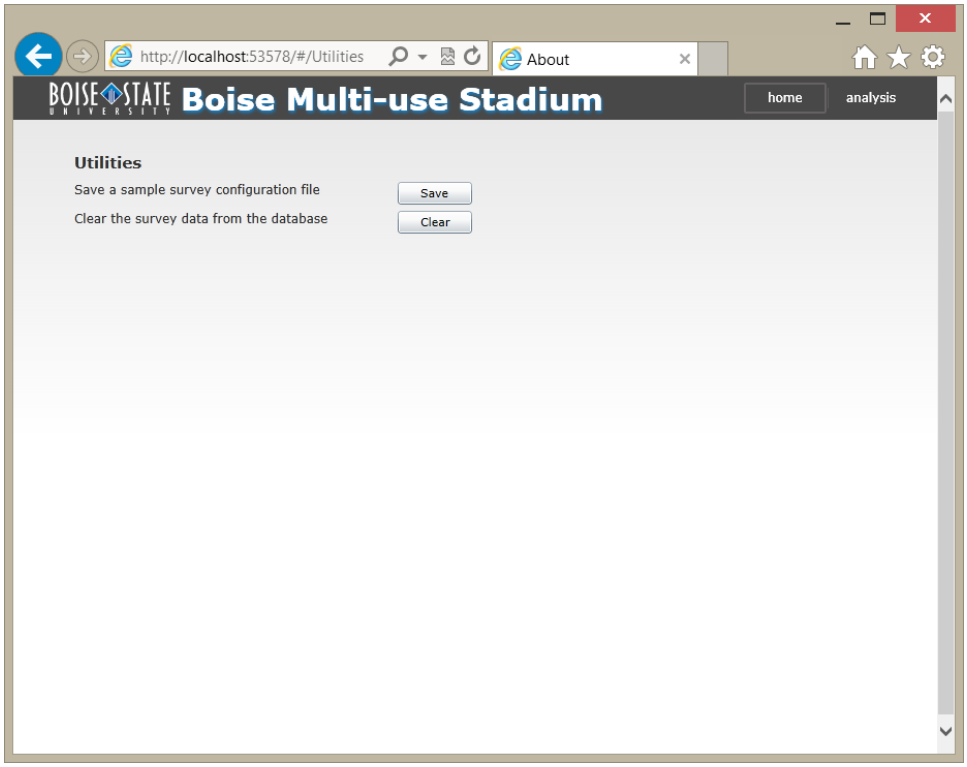

Figure 4.5: Utilities Screen

views in the data store as well as the returned dataset. This is a limitation to the application, but was adopted to provide an example of the possibilities of real-time analysis for a large dataset. Real-time analysis of an extremely large dataset cannot be performed in the abstract due to its inherent computational intensity. Providing a configuration-based mechanism for real-time analysis of the data being collected is an open question left to future research.

Utilities Screen The utilities screen (Figure 4.5) does not have a visible link from the other two pages. It is accessible by changing the URL to "/Utilities" after the hash tag.

There are two functions provided here for developer convenience. The first func-

tion allows for a sample configuration file to be generated from a survey that has been programmatically created. The second will remove all the data from the data store matching the survey identifier provided in the configuration.

4.4 Survey Data Retrieval

As has been previously stated, Microsoft SQL Server was selected for purposes of expediency for the pilot. However, the XML data stored in the table is readily accessible by using extensions to SQL specifically designed for this purpose. These extensions allow for XPATH statements to be executed to retrieve portions of the data contained in an XML field in the database. This is precisely the technique employed to retrieve the data for analysis. See Appendix B for the queries used to retrieve tabular data from the XML records used for the Analysis screen.

4.5 Solution Layout

The solution contains several projects. The purposes in partitioning the solution into multiple projects are to impose a modular framework suitable for future expansion on the code and make it more maintainable. For a complete listing of files in the project and their significance, see Appendix C.

SurveyEngine.Web This is the website for the project. It ultimately hosts the components compiled by the other projects. It also provides the SOAP service (in SurveyEngineService.svc) and a WCF RIA service (in AnalysisDomainService.cs and AnalysisDomainService.metadata.cs). The user accessible website (Default.aspx) is simply an HTML wrapper for the SurveyEngine Silverlight application.

SurveyEngine The survey engine is the main Silverlight application. It contains the framework for loading the configuration file and loadable modules, but does not depend upon them. It also contains references to the WCF RIA Service and SOAP service provided by SurveyEngine.web.

This assembly also contains the complete code for the analysis functionality of the application.

SurveyEngine.Types This module contains the data types used to create surveys.

SurveyEngine.Viewers This module contains the viewers associated with each of the types. These two are separated into different modules so that the viewers and types may be changed independently of each other.

SurveyEngine.Package This project is a deployable assembly containing the Types and Viewers module. Other than these references, there is no actual "code" contained in this assembly.

CHAPTER 5

RESULTS

Throughout the focus group pilot, datapoints were continuously collected from each user's computer. These datapoints were relayed through the server back to the analysis screen of each participant as well as the focus group facilitators. Although primary data was captured via the survey and computer model and secondary data from the census was also gathered for each participant, with only 8 participants, any patterns could indicate the usefulness of this technique, but will not be conclusive.

5.1 Quantitative Data Analysis

By observing the differences between the participants' responses at the end of each round, it becomes apparent that consensus was reached for at least some items. When deciding on the percentage of covered seating for the stadium, Figures 5.1, 5.2, and 5.3 show that although the participants started with very different ideas on what percentage of covered seating was best, and held on to those opinions through the second round, they were willing to come to a consensus for the third round.

After round one, there was no consensus in the number of seats. However, a warm discussion ensued between the participants between rounds one and two. During this discussion, regional comparisons and local competing facilities were compared. During the second round, a level of consensus, strongly indicative of a tipping-point

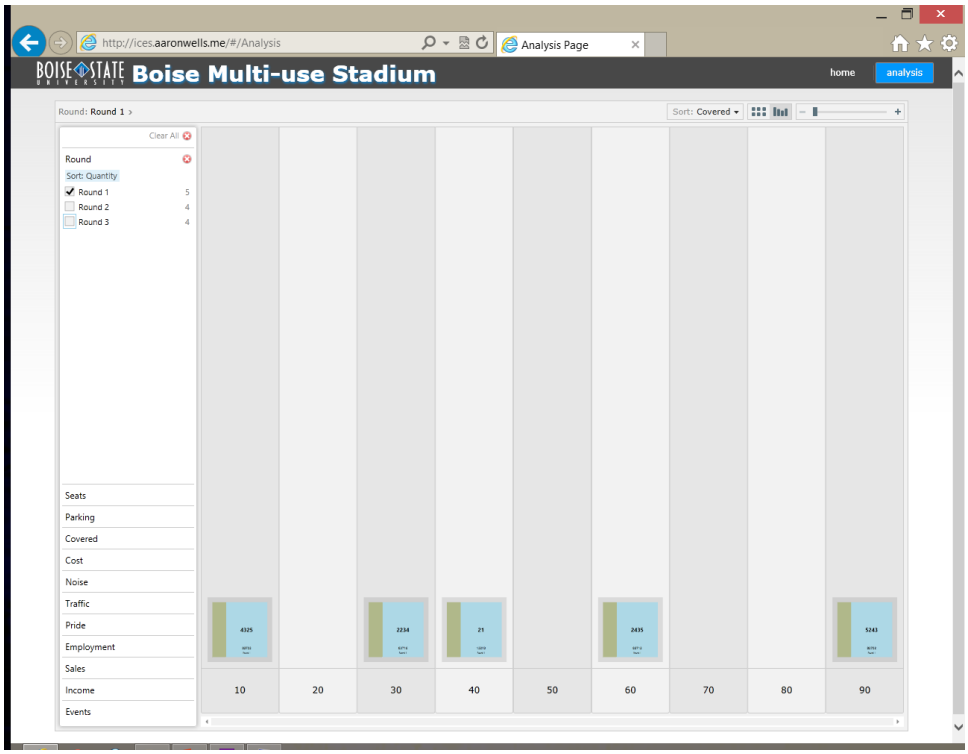

Figure 5.1: Round one percentage of covered seating

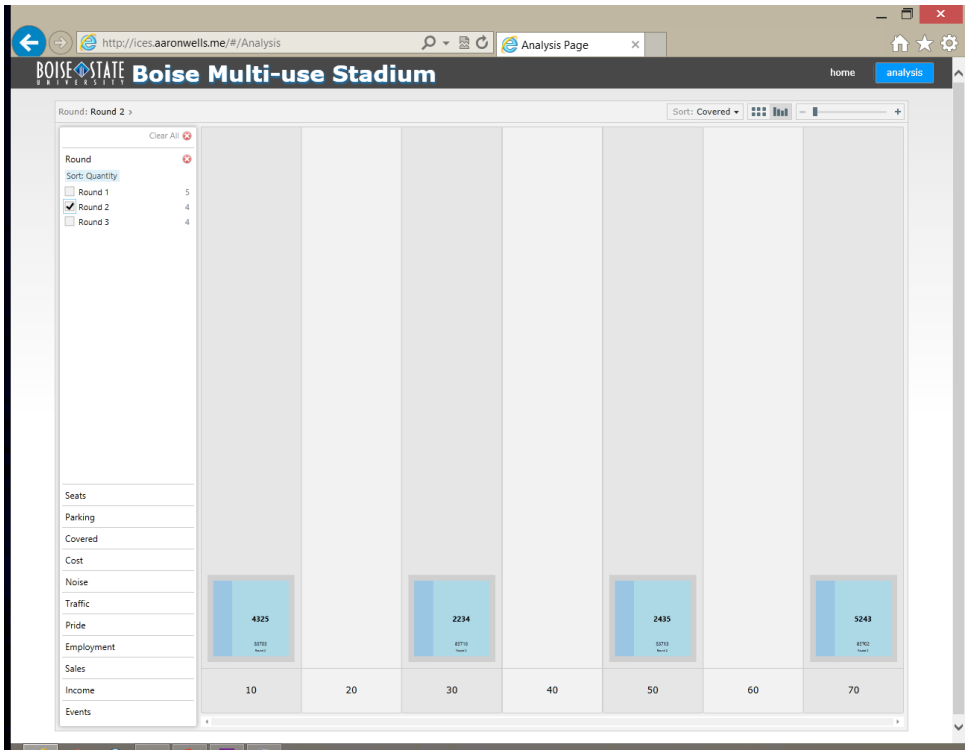

Figure 5.2: Round two percentage of covered seating

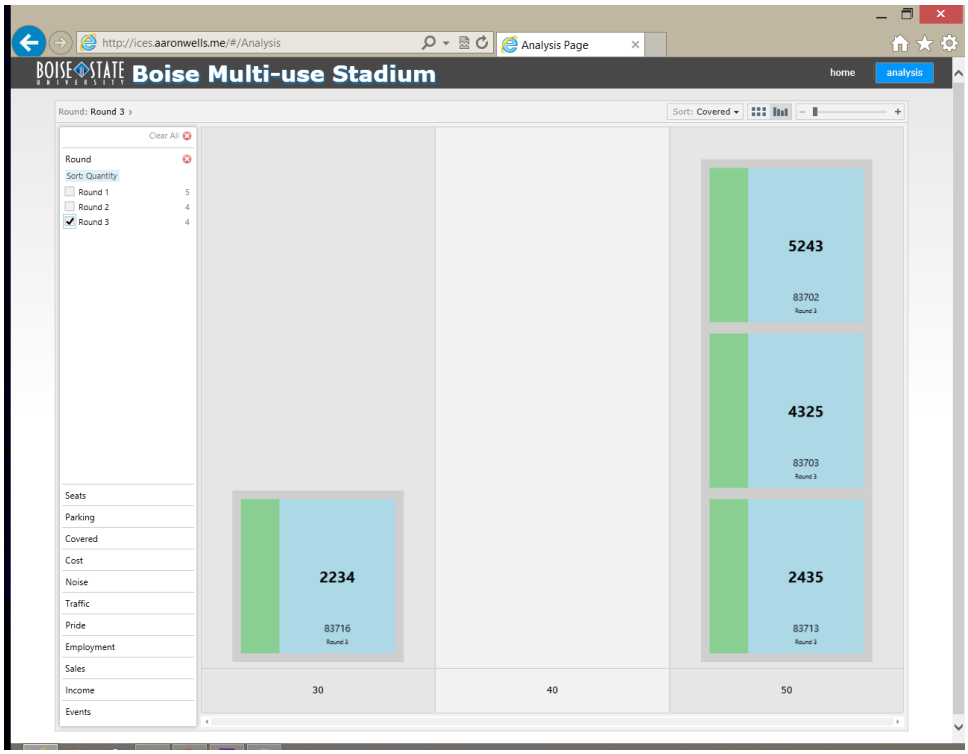

Figure 5.3: Round three percentage of covered seating

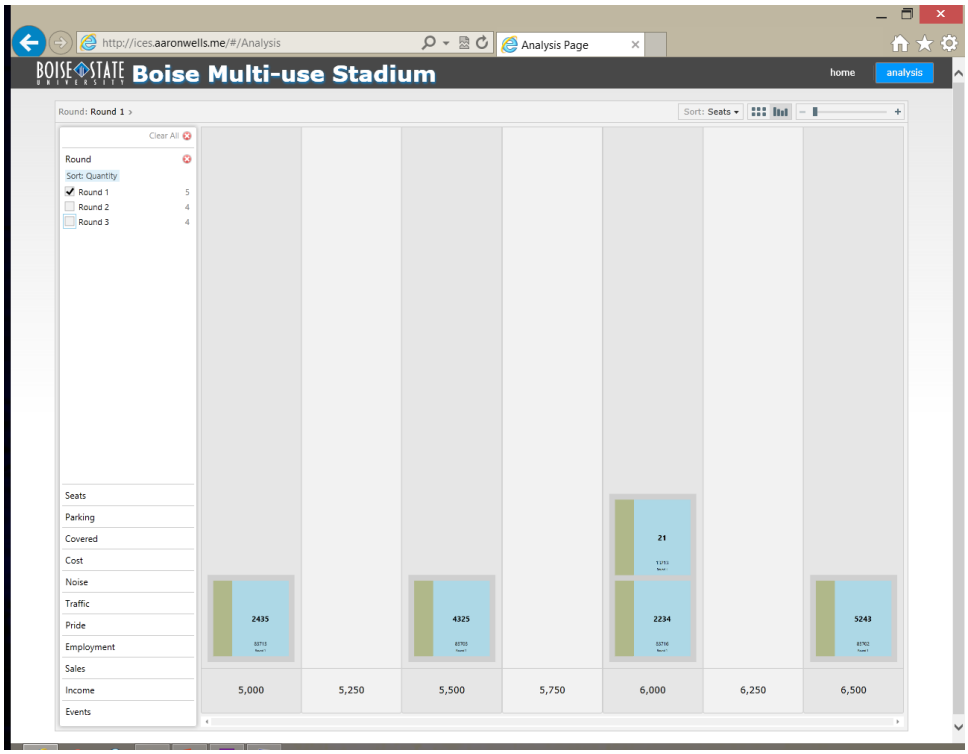

Figure 5.4: Round one number of seats

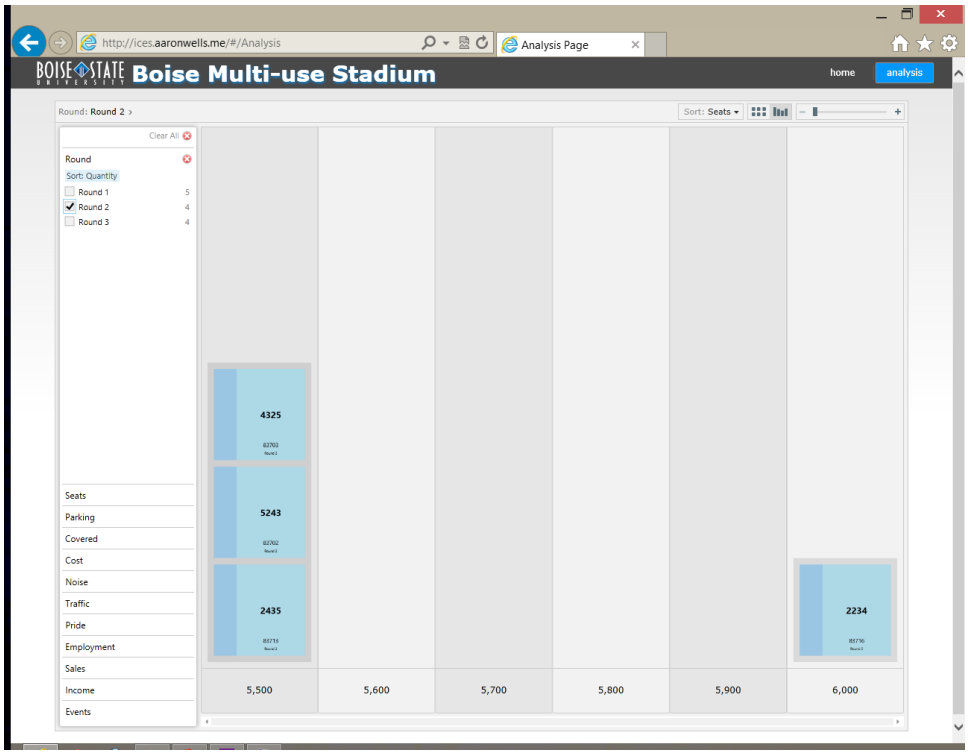

Figure 5.5: Round two number of seats

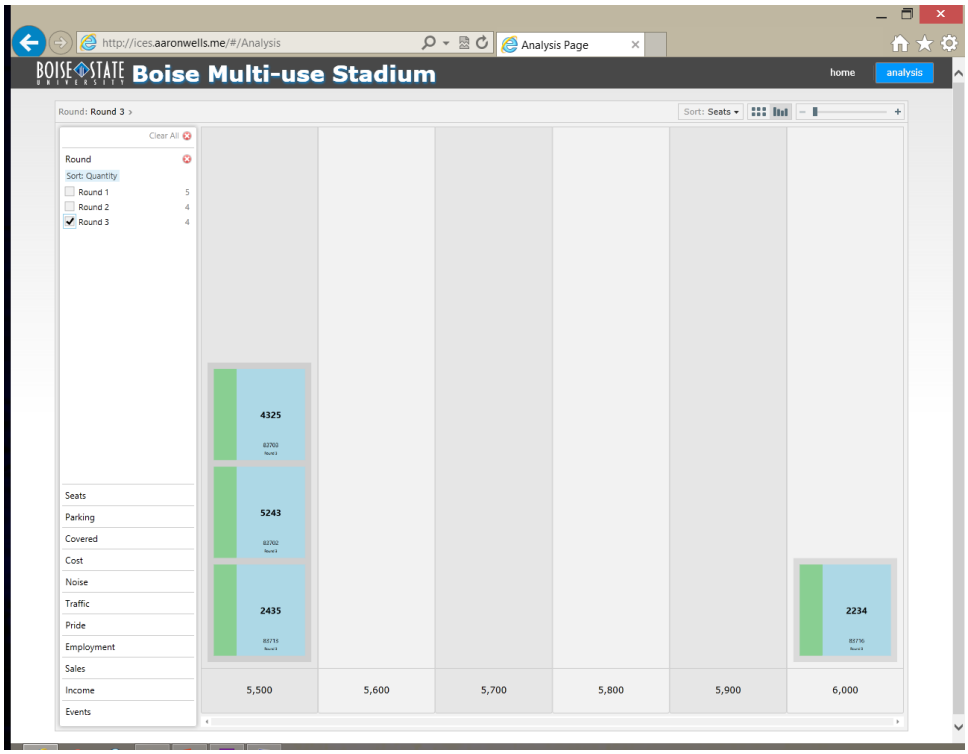

Figure 5.6: Round three number of seats

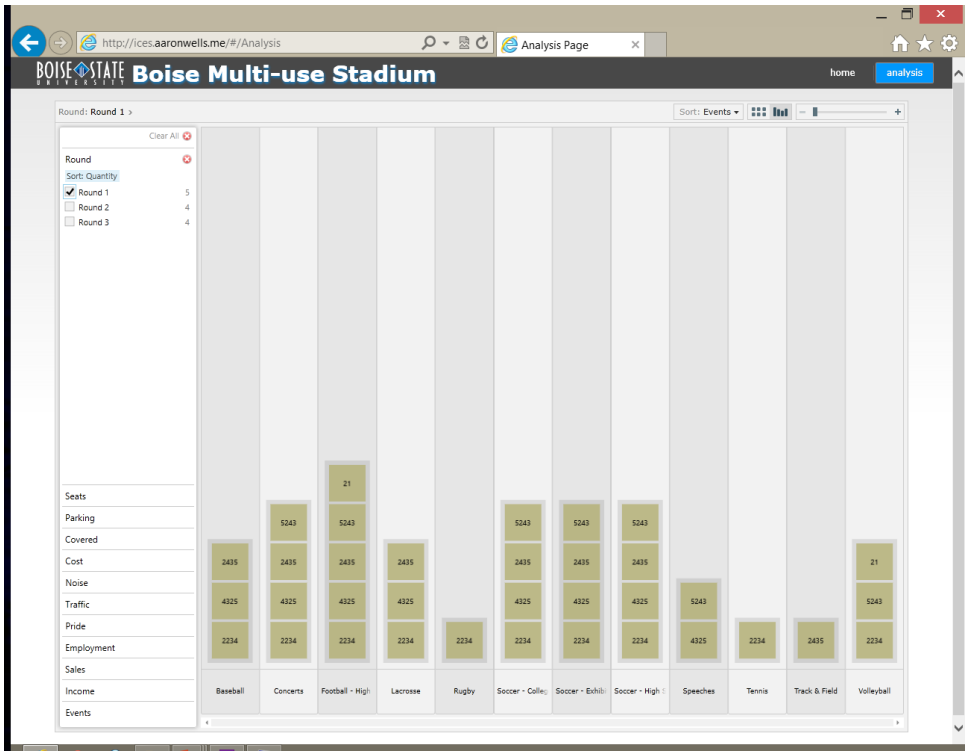

Figure 5.7: Round one selected events

was achieved. This result held constant through the third round for each participant. Figures 5.4, 5.5, and 5.6 show this progression.

Each participant had the ability to select as many of the events as they wished. Each event had an impact on the costs and benefits of the stadium. Given this dynamic, it may be presumed that more popular event types would receive the most votes; this appears to be the case with multiple events receiving three votes and only one receiving four in the first round 5.7. In subsequent rounds, however, three 5.8 and then five 5.9 events received unanimous support. While this does not clearly demonstrate a single tipping point, it does clearly show the underlying ability of individuals to act outside of their own personal interests for the perceived common desires.

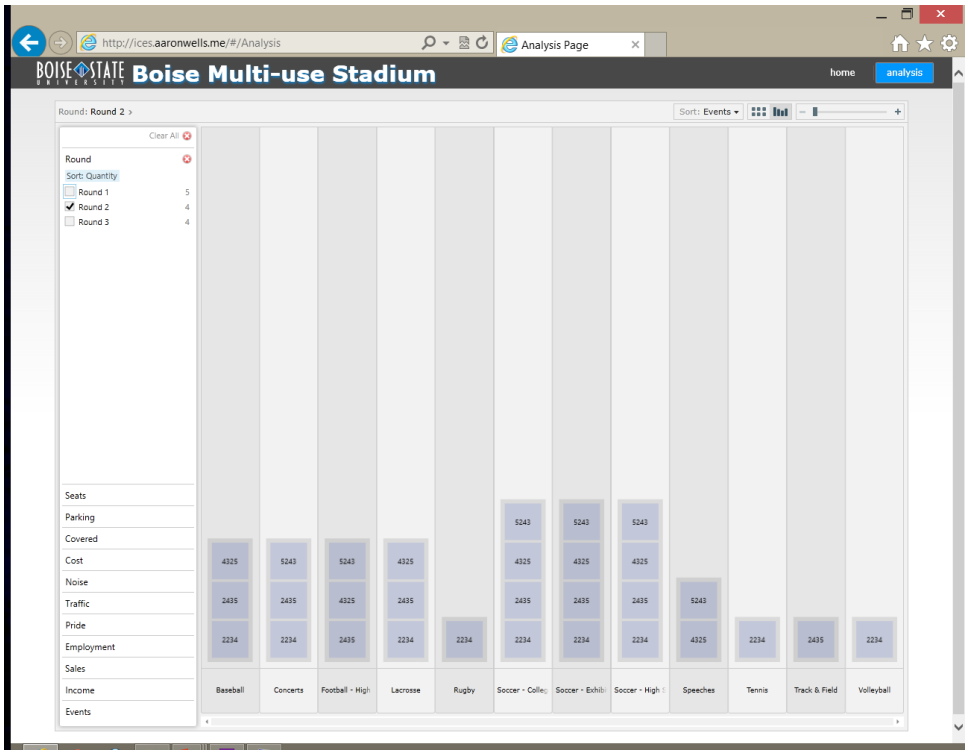

Figure 5.8: Round two selected events

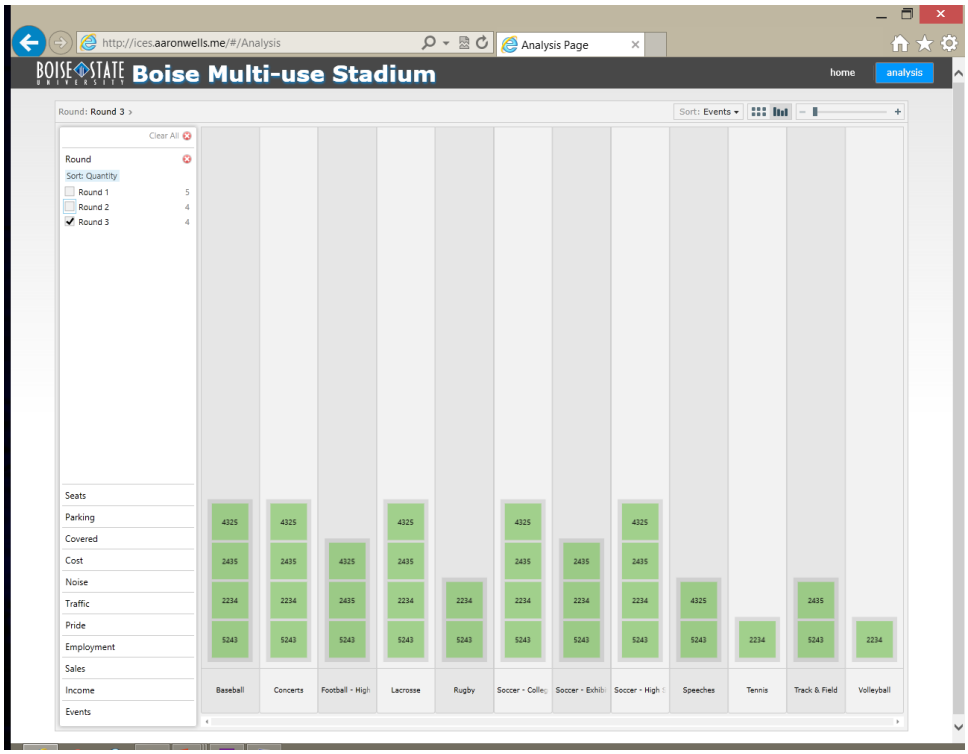

Figure 5.9: Round three selected events

Ultimately, the software framework appears to have worked quite well within the focus group to collect meaningful data, and allow observation of changes in the group dynamic during a deliberative process.

5.2 Qualitative Data Analysis

The qualitative data also indicated some interesting convergences. After the participants engaged in the decision making process they were debriefed by the study investigators. Specifically, each participant went to a debriefing room where they were individually asked to the following debriefing questions:

1. Whose opinion did you value the most? Why? On a scale of 1 to 10 how strongly did this person influence your outcomes?
2. Who else could be or was influenced by XX?
3. At what times were you tempted to change response but didn't and why?
4. At what times did you change your response and why?
5. What factors contributed to you reaching consensus or not with the rest of the group?
6. Identify two ways others influenced your actions/outcomes (position, knowledge etc.)?
7. Do you have social ties or organizational ties to any of the members on the session? Who and what are they (social, organizational)?
8. Was the technology you used today an effective tool to make decisions on economic development projects?

First, it was clear in each round that one person emerged as a thought leader. In the first round it was male who had expertise in professional sports and local economic development around a specific team. In the second session it was a female with expertise in downtown economic development. Participants in both groups indicated they were influenced because these people seemed the most knowledgeable. Five of the eight participants reported they were tempted to change their response by influences in the discussion, but did not. The actual factors that participants reported as causing them to change their responses including becoming familiar with the software, the group discussions, and trying different scenarios to maximize the economic impact. Six of the eight participants report either social or professional ties with at least one other person in their focus group. This may have played a role in some people being influenced by the two persons perceived as experts that emerged as thought leaders.

Finally, the participants unanimously agreed the software helped with their decision making because it allowed them to see the outcomes of their choices. Participants indicated they could see the software's applicability to other projects or issues. Some participants suggested it could be more sophisticated while others indicated a desire for something more intuitive. Their desired level of tool sophistication appeared to coincide with their familiarity with GIS or other modeling tools.

CHAPTER 6

CONCLUSIONS

6.1 Achievements

Any tool is subject to improvement and refinement from the moment it is first used. The focus group study and the software developed to be used for that study is no exception. It was pleasing to the author that the software framework was so well received both by the advisory committee and the participants of the focus group. The software framework succeeded in collecting and providing a meaningful analysis framework that provided enough of a validation of the "tipping point" hypothesis that additional study should be considered. The flexibility of the framework allowed for changes to the survey to be quickly implemented when changes to the focus group methodology were introduced.

As a platform for future work in community planning, and possibly areas of sociology or psychology, this platform, or some future variant could provide additional insights into many different types of social dynamics. It could be especially effective if the sessions were recorded with time stamp information. Time-stamped model interactions could then be tied to triggering external stimuli.

6.2 Future directions

There are many areas where the work performed for this project could be improved. Some of these were intentionally left out of scope due to time constraints, and others were realized as a result of our experience.

6.2.1 Pilot Methodology

The software framework was designed in such a way as to collect data in real time as participants modify their choices. This advance in data collection capability represents the potential to continuously collect information from participants throughout a decision-making process rather than at arbitrary and externally imposed data collection points such as were used for this study.

If this data stream were augmented with a transcript of the discussion between participants, the data *may* provide additional insights into the individual and group decision making process. It is possible that individuals experience a personal tipping point distinct from that experienced by the group.

The framework was also designed to run across the public Internet. In this way, larger numbers of individuals could simultaneously (or even asynchronously) express their preferences based upon a reaction to a live or recorded discussion, if not participate in the discussion personally.

The framework could also be used to inform each participant of the consensus (or lack thereof) of the population. We hypothesize that this feedback loop would also effect the decision process.

6.2.2 Communication Efficiency

The means of collecting model data in the pilot are naive. They collect a data point every 10 seconds from each user. The analysis portion of the application also polls the server for new data every 10 seconds; for a total potential latency of 20 seconds.

Much work could be done to optimize the communication efficiency of the application. The following are a few suggestions:

1. Use a broadcast protocol between clients on the same network segment from super nodes to other clients.

2. Send model data to the server only when there have been changes made to the model.

3. Send a timestamp token to the server indicating the freshness of the data and only fetch data which is newer than the timestamp.

4. Compress the data.

5. Notify clients rather than using polling. The new "web sockets" capabilities introduced in HTML5 may provide this ability.

6.2.3 Immersive models

The pilot project used primitive feedback to users indicating noise levels around a theoretical stadium. This feedback allowed participants to interact in a more immersive way than a simple number could provide. Other immersive models employing sound, three-dimensional models, and other "game like" capabilities could encourage users to experiment and discover solutions not envisioned by the model creators.

Future work may expand upon this theory, providing more efficient means of feedback, more interactive or immersive models, asynchronous group model interactions, and less intrusive data collection mechanisms.

REFERENCES

[1] Bugs, Geisa, Carlos Granell, Oscar Fonts, Joaquin Huerta, and Marco Painho. 2010. "An assessment of Public Participation GIS and Web 2.0 technologies in urban planning practice in Canela, Brazil." Cites 172-181.

[2] Bugnion, Laurent. n.d. MVVM Light Toolkit. Accessed March 16, 2013. http://www.galasoft.ch/mvvm/.

[3] ESRI. n.d. Mapping APIs and Services for Developers. Accessed March 16, 2013. http://www.esri.com/getting-started/developers/get-started/silverlight.

[4] Gladwell, M. 2006. "The tipping point: How little things can make a big difference." By Brown and Little.

[5] Kocabas, Verda, Suzana Dragivcevic, and Eugene McCann. 2012. "Integration of a GIS-Bayesian Network Agent-based Model in a Planning Support System as a Framework for Policy Generation." Journal of the Urban and Regional Information Systems Association 35-52.

[6] Microsoft. n.d. Silverlight PivotViewer. Accessed March 16, 2013. http://www.microsoft.com/silverlight/pivotviewer/.

[7] Rezaei, Golriz, and Michael Kirley. 2009. "The effects of time-varying rewards on the evolution of cooperation." Evol. Intel. 207-218.

[8] Sosis, Richard, Sharon Feldstein, and Kim Hill. 1997. "Bargaining Theory and Cooperative Fishing Participation on Ifaluk Atoll." Human Nature, October 15: 163-203.

[9] Tang, Kathy X., and Nigel M. Waters. 2005. "The internet, GIS and public participation in transportation planning." Progress in Planning 7-62.

[10] Zellner, Moira L., Leilah B. Lyons, Charles J. Hoch, Jennifer Weizerorick, Carl Kunda, and Daniel C. Milz. 2012. "Modeling, Learning, and Planning Together: An Application of Participatory Agent-based Modeling to Environmental Planning." Journal of the Urban and Regional Information Systems Association 77-92.

APPENDIX A

SAMPLE SURVEY CONFIGURATION FILE

```
<?XML version="1.0" encoding="utf-8"?>
<SurveyConfiguration XMLns:xsi="http://www.w3.org/2001/XMLSchema-instance"
XMLns:xsd="http://www.w3.org/2001/XMLSchema">
 <Title>Sample Survey</Title>
 <Rounds>
 <Round>
 <Id>1</Id>
 <Name>Demo Round</Name>
 <Description>This round does absolutely nothing</Description>
 <Recorded>false</Recorded>
 <AllowedTime>0</AllowedTime>
 </Round>
 </Rounds>
 <Pages>
 <Page>
 <Title>Introduction</Title>
 <VisibleInRounds>
 <int>1</int>
 <int>2</int>
 </VisibleInRounds>
 <Sections>
 <Section>
 <WrappedObject xsi:type="MultipleChoiceQuestionOfStringInt32">
 <Title>Pick Something</Title>
 <Choices>
 <ChoiceOfStringInt32>
 <Title>apple</Title>
 <Value>1</Value>
 </ChoiceOfStringInt32>
 <ChoiceOfStringInt32>
 <Title>banana</Title>
 <Value>2</Value>
```

```
</ChoiceOfStringInt32>
<ChoiceOfStringInt32>
<Title>cherry</Title>
<Value>3</Value>
</ChoiceOfStringInt32>
<ChoiceOfStringInt32>
<Title>donut</Title>
<Value>4</Value>
</ChoiceOfStringInt32>
</Choices>
<Value>0</Value>
</WrappedObject>
</Section>
<Section>
<WrappedObject xsi:type="LabeledValueOfInt32">
<Title>Type an integer</Title>
<Value>0</Value>
</WrappedObject>
</Section>
<Section>
<WrappedObject xsi:type="LabeledValueOfString">
<Title>Describe how you first meet Johnny?</Title>
</WrappedObject>
</Section>
</Sections>
</Page>
<Page>
<Title>Stadium</Title>
<Sections>
<Section>
<WrappedObject xsi:type="StadiumModel">
<AreaPerSeat>0</AreaPerSeat>
<AreaPerParking>0</AreaPerParking>
<PropertyCost>0</PropertyCost>
<CostPerSeat>0</CostPerSeat>
<ParkingSurfaceCost>0</ParkingSurfaceCost>
<ParkingElevatedCost>0</ParkingElevatedCost>
<HomeownerExemption>0</HomeownerExemption>
<Contributions>0</Contributions>
<InterestRate>0</InterestRate>
<AmortizationPeriod>0</AmortizationPeriod>
<AnnualOperatingCosts>0</AnnualOperatingCosts>
<SessionId>776ac7fb-b3dd-4a5b-930a-e713e5ea2162</SessionId>
```

```
<AvailableArea>1000000</AvailableArea>
<TaxBase>20000000</TaxBase>
<Events>
<Event>
<Id>0</Id>
<Name>Football</Name>
<Dimension1>300</Dimension1>
<Dimension2>60</Dimension2>
<CostMultiplier>1.1</CostMultiplier>
<Noise>90</Noise>
<Attendance>0</Attendance>
<Vehicles>0</Vehicles>
<Speed>0</Speed>
<Pride>0</Pride>
<RevenuePerPerson>0</RevenuePerPerson>
<EventsPerYear>0</EventsPerYear>
<StadiumRental>0</StadiumRental>
<StadiumPercentage>0</StadiumPercentage>
<Selected>false</Selected>
</Event>
<Event>
<Id>0</Id>
<Name>Soccer</Name>
<Dimension1>330</Dimension1>
<Dimension2>90</Dimension2>
<CostMultiplier>1.1</CostMultiplier>
<Noise>80</Noise>
<Attendance>0</Attendance>
<Vehicles>0</Vehicles>
<Speed>0</Speed>
<Pride>0</Pride>
<RevenuePerPerson>0</RevenuePerPerson>
<EventsPerYear>0</EventsPerYear>
<StadiumRental>0</StadiumRental>
<StadiumPercentage>0</StadiumPercentage>
<Selected>false</Selected>
</Event>
</Events>
<MapEnvelope>
<XMin>-12963361.5466758</XMin>
<YMin>5396771.91209996</YMin>
<XMax>-12924225.7881938</XMax>
<YMax>5413912.76755914</YMax>
```

```
<ZMin>NaN</ZMin>
<ZMax>NaN</ZMax>
<MMin>NaN</MMin>
<MMax>NaN</MMax>
</MapEnvelope>
<Seats>10000</Seats>
<Parking>3000</Parking>
<Covered>0</Covered>
<PropertyValue>0</PropertyValue>
</WrappedObject>
</Section>
</Sections>
</Page>
<Page>
<Title>Locations</Title>
<Sections>
<Section>
<WrappedObject xsi:type="MultipleChoiceQuestionOfStringGeospatialLocation">
<Title>Where should we put the new stadium?</Title>
<Choices>
<ChoiceOfStringGeospatialLocation>
<Title>Downtown Location</Title>
<Value>
<Location>192.34,123.33</Location>
</Value>
</ChoiceOfStringGeospatialLocation>
<ChoiceOfStringGeospatialLocation>
<Title>Boise Hawks Field</Title>
<Value>
<Location>191.34,123.34</Location>
</Value>
</ChoiceOfStringGeospatialLocation>
<ChoiceOfStringGeospatialLocation>
<Title>Airport View</Title>
<Value>
<Location>192.34,123.25</Location>
</Value>
</ChoiceOfStringGeospatialLocation>
<ChoiceOfStringGeospatialLocation>
<Title>West End</Title>
<Value>
<Location>192.34,123.33</Location>
</Value>
```

```
 </ChoiceOfStringGeospatialLocation>
 </Choices>
 </WrappedObject>
 </Section>
 </Sections>
 </Page>
 <Page>
 <Title>3d Model</Title>
 <Sections>
 <Section>
 <WrappedObject xsi:type="Obj3dFile">
 <Filename>foobar.obj</Filename>
 </WrappedObject>
 </Section>
 </Sections>
 </Page>
 </Pages>
 <SurveyUid>9a38a9c5-8938-486d-bfbc-5225f01285fc</SurveyUid>
 <InstanceUid>21f551d2-17ff-47aa-bdfa-0a2866e6c0e2</InstanceUid>
</SurveyConfiguration>
```

APPENDIX B

XML SQL QUERIES

Query for vSurvey

```
SELECT
 ISNULL(M.Id,0) Id,
 SurveyData.value('(/Survey//InstanceUid)[1]', 'UniqueIdentifier') AS
InstanceUid,
 SurveyData.value('(/Survey//SurveyUid)[1]', 'UniqueIdentifier') AS SurveyUid,
 SurveyData.value('(/Survey//Round)[1]', 'nvarchar(max)') AS Round,
 SurveyData.value('(/Survey//Closed)[1]', 'bit') AS Closed,
 SurveyData.value('(/Survey//ElapsedTime)[1]', 'int') AS ElapsedTime,
 SurveyData.value('(/Survey//AbsoluteTime)[1]', 'DateTime') AS AbsoluteTime,
 SurveyData.value('(/Survey//Value[@Index="1"])[1]', 'nvarchar(max)') AS
SurveyKey,
 SurveyData.value('(/Survey//Value[@Index="3"])[1]', 'nvarchar(max)') AS
ZipCode,
 SurveyData.value('(/Survey//StadiumModel//Parking)[1]', 'int') AS Parking,
 SurveyData.value('(/Survey//StadiumModel//Covered)[1]', 'int') AS Covered,
 SurveyData.value('(/Survey//StadiumModel//Cost)[1]', 'float') AS Cost,
 SurveyData.value('(/Survey//StadiumModel//Noise)[1]', 'float') AS Noise,
 SurveyData.value('(/Survey//StadiumModel//Traffic)[1]', 'float') AS Traffic,
 SurveyData.value('(/Survey//StadiumModel//Pride)[1]', 'float') AS Pride,
 SurveyData.value('(/Survey//StadiumModel//Employment)[1]', 'float') AS
Employment,
 SurveyData.value('(/Survey//StadiumModel//Sales)[1]', 'float') AS Sales,
 SurveyData.value('(/Survey//StadiumModel//Income)[1]', 'float') AS Income,
 (SELECT SUBSTRING( CONVERT(NVARCHAR(max),
 S.SurveyData.query('for $i in //Event return
concat(",",data($i/@Name))')),2,5000 ))
   as Events
FROM     dbo.Survey s
JOIN (Select Min(Id) Id, Max(Id) MaxId, [Uid] from Survey Group By [Uid]) M
 ON S.Id = M.MaxId --and S.Uid = M.Uid
WHERE SurveyData.value('(/Survey//Value[@Index="1"])[1]', 'nvarchar(max)') <>
```

,,

Query for vSurveyEvent

```
Select
ISNULL(ROW_NUMBER() OVER (ORDER BY s.Id, T2.ev.value('(.)[1]','int')),0)
as Id,
ISNULL(M.Id,0) AS SurveyId,
[Uid] AS InstanceUid,
T2.ev.value('(.)[1]','int') as EventId,
T2.ev.value('(@Name)[1]','nvarchar(max)') as EventName
From dbo.Survey s
join (Select Min(Id) Id, Max(Id) MaxId from Survey Group By [Uid]) M
on S.Id = M.MaxId
CROSS APPLY SurveyData.nodes('/Survey/StadiumModel/Events/Event') as T2(ev)
```

APPENDIX C

SOLUTION FILES

|SurveyEngine.sln	The solution file
`+---.nuget`	Contains package info
+—packages	Contains package cache
+—SurveyEngine	Contains main Silverlight project
||App.xaml	XAML for application
||App.xaml.cs	Code behind for application
||MainPage.xaml	XAML for main application page
||MainPage.xaml.cs	Code behind for main page
||packages.config	Packages needed by project
||ServiceReferences.ClientConfig	
||SurveyEngine.csproj	Project file
||SurveyEngine.csproj.user	Project user settings
|+—Assets	
||Styles.xaml	Styling information
|+—Properties	Settings files for project
||AppManifest.XML	
||AssemblyInfo.cs	Generated project information
||InBrowserSettings.XML	

||OutOfBrowserSettings.XML	
|+—Service References	Configuration and generated reference information
||\—SurveyEngineServiceReference	
||configuration.svcinfo	
||configuration91.svcinfo	
||Reference.cs	
||Reference.svcmap	
||SurveyEngineService.disco	
||SurveyEngineService.wsdl	
||SurveyEngineService.xsd	
||SurveyEngineService1.wsdl	
||SurveyEngineService1.xsd	
|+—SurveyEngineServiceReference	
||SurveyEngineServiceClient.cs	
|+—UIHelpers	Classes to help Silverlight
||ColorConverter.cs	Converts data to different colors
||TabConverter.cs	Converts Page to tabs on UI
||VisibilityConverter.cs	Hides or shows data based on round
|+—ViewModel	
||AnalysisViewModel.cs	VM for analysis screen
||MainViewModel.cs	VM for home screen
||ViewModelLocator.cs	Helps in MVVM binding

|+—Views	
||Analysis.xaml	XAML for analysis page
||Analysis.xaml.cs	Code behind
||ErrorWindow.xaml	XAML for error screen
||ErrorWindow.xaml.cs	Code behind
||Home.xaml	XAML for home screen
||Home.xaml.cs	
||RoundWindow.xaml	XAML for round popup screen
||RoundWindow.xaml.cs	
||Utilities.xaml	XAML for utilities page
||Utilities.xaml.cs	
|\—Web	Extensions to RIA generated classes
|AnalysisDomainContext.cs	
|vSurvey.cs	
+—SurveyEngine.DB	Project for database definitions
+—SurveyEngine.Package	Deployable package for Types
||SurveyEngine.Package.csproj	
|\—Properties	
|AppManifest.XML	
+—SurveyEngine.Types	Project with components
||ExportFactoryAttribute.cs	Attribute defining exported components
||ISurveyData.cs	Interface definition for data
||ISurveyDataView.cs	Interface definition for data presentation

||IUserControlFactory.cs	Interface definition for creating user controls
||packages.config	List of packages
||StadiumModel.cs	Stadium model type
||SurveyConfiguration.cs	Configuration types
||SurveyEngine.Types.csproj	Project file
||SurveyEngine.Types.csproj.user	Project user settings
||SurveyTypes.cs	Other basic survey types
|\—Properties	Project properties
+—SurveyEngine.Web	Website
||AnalysisDomainService.cs	WCF RIA service for analysis
||AnalysisDomainService.metadata.cs	Additional data for analysis service
||AnalysisModel.Designer.cs	Visual configuration of model
||AnalysisModel.edmx	Analysis data model
||AnalysisModel.edmx.diagram	Configuration information
||Compare.aspx	Simple survey host page
||Default.aspx	Default survey host page
||packages.config	List of packages
||Silverlight.js	Needed by Silverlight
||SurveyEngine.Web.csproj	Project file
||SurveyEngine.Web.csproj.user	Project user configuration
||SurveyEngineService.svc	Survey Engine SOAP service
||SurveyEngineService.svc.cs	Code behind
||Web.config	Baseline website configuration

||Web.Debug.config	Changes for debug
||Web.Release.config	Changes for release
|+—App_Code	Empty
|+—App_Data	Empty
|+—bin	Target for compile
|+—ClientBin	Target for Silverlight components
||CompareConfiguration.XML	Configuration files for above host pages
||DemoConfiguration.XML	This one too
||logo.png	BSU logo loaded by Silverlight
||SurveyEngine.Package.xap	Extensions to Silverlight library
||SurveyEngine.xap	Main Silverlight app
|\—Properties	
\—Viewers	Project containing UI for types
|FreeformIntViewer.xaml	Viewer for integer inputs
|FreeformIntViewer.xaml.cs	
|FreeformTextViewer.xaml	Viewer for text inputs
|FreeformTextViewer.xaml.cs	
|GeospatialViewer.xaml	Viewer for ESRI map data
|GeospatialViewer.xaml.cs	
|MultipleChoiceViewer.xaml	Viewer for Multiple Choice inputs
|MultipleChoiceViewer.xaml.cs	
|Obj3dViewer.xaml	Viewer for 3d models
|Obj3dViewer.xaml.cs	

|packages.config	List of packages
|SnappingSlider.cs	Custom snapping slider control
|StadiumModelViewer.xaml	Viewer for stadium model
|StadiumModelViewer.xaml.cs	
|SurveyEngine.Viewers.csproj	Project file
|SurveyEngine.Viewers.csproj.user	Project user settings
|SurveyEventViewer.xaml	Viewer for survey events
|SurveyEventViewer.xaml.cs	
+—Properties	
|AssemblyInfo.cs	Generated file
\—Service References	